Spektakuläre Android-Apps entwickeln

1

In diesem Kapitel

- Gründe für die Entwicklung von Android-Apps erkennen
- Erste Schritte in der Android-Programmierung unternehmen
- Mit der Hardware arbeiten
- Sich mit der Software vertraut machen

Google ist Spitze! Google übernahm 2005 die Android-Plattform (siehe Kasten »Die Wurzeln von Android« weiter hinten in diesem Kapitel), um ein Betriebssystem für Mobilgeräte zu entwickeln und es als offene Plattform zur Verfügung zu stellen. Google investiert weiterhin Zeit und Ressourcen in das Android-Projekt. Obwohl es die Geräte erst seit Oktober 2008 gibt, wurden mittlerweile über eine Milliarde Android-Geräte aktiviert, und täglich kommen mehr als eine Million hinzu. Innerhalb weniger Jahre konnte Android *riesigen* Einfluss gewinnen!

Nie war es für Android-Entwickler einfacher, Geld mit der App-Entwicklung zu verdienen. Android-Anwender vertrauen Google. Und da Ihre App im *Google Play Store* angeboten wird, sind auch viele Anwender bereit, Ihrer App zu vertrauen.

Die Wurzeln von Android

Es ist nur wenig bekannt, dass Google das Android-Projekt nicht ins Leben gerufen hat. Anfangs wurde das Betriebssystem Android von einem kleinen Start-up-Unternehmen in Silicon Valley namens *Android, Inc.* entwickelt, das dann im August 2005 von Google übernommen wurde. Die Unternehmensgründer kamen aus verschiedenen Technologieunternehmen des Internet-Bereichs, wie zum Beispiel *Danger*, *Wildfire Communications, T-Mobile* und *WebTV*. Google übernahm sie mit in das Google-Team, um dort das heute umfassende Android-Betriebssystem für Mobilgeräte zu entwickeln.

Warum für Android entwickeln?

Die Frage sollte eigentlich »Warum *nicht* für Android entwickeln?« lauten. Wollen Sie, dass Ihre App weltweit für Millionen Anwender verfügbar ist? Wollen Sie Apps sofort nach Abschluss der Entwicklung und Tests veröffentlichen? Gefällt es Ihnen, für offene Plattformen zu entwickeln? Wenn Sie eine dieser Fragen bejaht haben, dürften Sie die Antwort kennen. Sollten Sie aber noch unentschlossen sein, lesen Sie einfach weiter.

Marktanteil

Als Entwickler können Sie Apps für einen boomenden Markt entwickeln. Die Anzahl der genutzten Android-Geräte ist größer als die Anzahl aller Geräte unter anderen Betriebssystemen für Mobilgeräte zusammen. Über den *Google Play Store* gelangt Ihre App direkt und einfach zum Anwender. Die Anwender müssen nicht das Internet nach zu installierenden Apps durchsuchen. Sie müssen nur den auf ihren Geräten vorinstallierten Google Play Store nutzen, um an alle *Ihre* Apps zu kommen. Da der Google Play Store auf den meisten Android-Geräten vorinstalliert ist (ein paar Ausnahmen sind in Kapitel 19 aufgelistet), suchen Anwender üblicherweise erst einmal im Google Play Store nach von ihnen benötigten Apps. Es ist daher gar nicht so ungewöhnlich, wenn die Anzahl der Downloads einer App binnen weniger Tage förmlich explodiert.

Zeit für die Vermarktung

Angesichts all der APIs (Anwendungsprogrammierschnittstellen – Application Programming Interfaces), die sich mit im Lieferumfang von Android befinden, lassen sich innerhalb relativ kurzer Zeit umfassende Anwendungen entwickeln. Nach der Registrierung beim Google Play Store müssen Sie Ihre Apps nur noch hochladen und veröffentlichen. Im Unterschied zu den Marktplätzen anderer Mobilgeräte gibt es beim Google Play Store keine Genehmigungsverfahren für Apps. Sie müssen Ihre Apps nur entwickeln und veröffentlichen.

Grundsätzlich kann jeder Apps veröffentlichen, es dürfte aber besser sein, wenn Sie sich an die Google-Bedingungen halten und Ihre Apps familienfreundlich gestalten. Vergessen Sie nicht, dass Android-Anwender aus den verschiedensten Regionen der Welt und allen Alterskategorien kommen.

Offene Plattform

Das Android-Betriebssystem ist eine *offene Plattform* und damit an keinen Hardwarehersteller und/oder Anbieter gebunden. Wie man sich leicht vorstellen kann, konnte Android durch seine freie Verfügbarkeit schnell Marktanteile gewinnen. Nichts hält Sie davon ab, sich den Android-Quellcode anzusehen. Sie können ihn über `https://source.android.com` herunterladen. Durch den quelloffenen Code können Hersteller eigens angepasste Benutzeroberflächen oder auch Bedienoberflächen (UI – User Interface) erstellen und sogar neue Funktionen für bestimmte Geräte hinzufügen.

Gerätekompatibilität

Android läuft auf vielen Geräten mit unterschiedlichen Bildschirmabmessungen und Auflösungen, wie beispielsweise Uhren, Handys, Tablets, Fernsehgeräten und anderen. Zudem enthält es die Werkzeuge zur Entwicklung von Apps, die verschiedene Gerätetypen unterstützen. Wenn Ihre App beispielsweise nur mit einer Kamera an der Vorderseite des Geräts funktioniert, wird sie im Google Play Store nur bei Benutzung entsprechend ausgestatteter Geräte angezeigt. Diese Art der Hardwareerkennung wird bei Android *Feature Detection* genannt. (Mehr zur Veröffentlichung Ihrer Apps im Google Play Store erfahren Sie in Kapitel 8.)

Kombinierbarkeit (Mashups)

Wenn Sie zwei oder mehr Dienste kombinieren, um eine App zu erstellen, wird dies *Mashup* genannt. Sie können beispielsweise ein Mashup erstellen, wenn Sie die Kamera und die Ortungsdienste von Android nutzen, um ein Foto aufzunehmen, in dem die genaue Positionsangabe angezeigt wird. Oder Sie können die Map-API mit der Kontaktliste kombinieren, um all Ihre Kontakte in einer Landkarte anzeigen zu lassen. Die folgenden Mashup-Beispiele sollen Ihre Fantasie noch ein wenig stärker anregen:

- ✔ **Geolokation und soziale Netze:** Nehmen wir an, Sie wollen eine App schreiben, die Ihren aktuellen Standort mittels geologischer Ortsbestimmung *(Geolokation)* über den gesamten Tag hinweg alle zehn Minuten auf *Twitter* meldet. Kein Problem. Dazu müssen Sie nur die Lokalisierungsdienste von Android und die Twitter-API eines Drittanbieters (zum Beispiel *iTwitter*) miteinander kombinieren.
- ✔ **Geolokation und Spiele:** Ortsabhängige Spiele werden immer beliebter und bieten tolle Möglichkeiten, Anwender in Spiele einzubinden. Ein Spiel könnte als Hintergrunddienst laufen, den aktuellen Standort eines Spielers feststellen und diesen dann mit den Standorten anderer Anwender in derselben Gegend vergleichen. Befindet sich dann beispielsweise ein anderer Anwender in weniger als einem Kilometer Entfernung, könnte der erste Spieler benachrichtigt werden und ihn zum (Spiele-)Kampf herausfordern. Ermöglicht wird dies durch die leistungsfähigen Technologien von GPS und Android. Falls Sie daran interessiert sind, Spiele für Android zu entwickeln, lesen Sie unter `https://developers.google.com/games/services/` nach. Dort finden Sie weitere Informationen über die Google-Play-Games-Services.
- ✔ **Kontakte und Internet:** Angesichts der Vielzahl der verfügbaren nützlichen APIs lassen sich über die Kombination von Funktionen mehrerer APIs leicht umfassende Apps erstellen. Sie können beispielsweise die Namen aus Kontaktlisten und das Internet gemeinsam dazu benutzen, um Grußkarten-Apps zu erstellen. Sie könnten den Anwendern über Ihre App auch die Kontaktaufnahme mit sich erleichtern oder ihnen die Weiterleitung der App an Freunde ermöglichen. All dies ist mit den integrierten APIs machbar. (Mehr über APIs erfahren Sie im Abschnitt »Google-APIs« weiter hinten in diesem Kapitel.)

Entwicklern bietet Android nahezu grenzenlose Möglichkeiten, mit denen Sie jedoch vorsichtig umgehen sollten. Bilden Sie sich selbst ein möglichst objektives Urteil, bevor Sie Ihre Apps veröffentlichen und der Masse zugänglich machen. Dass Ihnen als Hintergrundmotiv dieses Filmchen gefällt, in dem Sie auf Ihrer Geburtstagsparty den Hula tanzen, bedeutet nicht zwangsläufig, dass andere das auch sehen wollen oder sehen sollten.

Grundlagen der Android-Programmierung

Sie müssen glücklicherweise nicht Mitglied des Mensa-Clubs sein, um Android-Apps programmieren zu können. Dass *Java* als Standardsprache dient, erleichtert die Android-Programmierung. Aber auch wenn sich Android-Programme relativ leicht schreiben lassen, ist der Programmcode im Allgemeinen nicht unbedingt ein Kinderspiel.

Wenn Sie bisher noch nie Apps entwickelt haben, bildet dieses Buch vielleicht nicht den optimalen Einstieg. Zum Erlernen der Grundlagen empfehle ich Ihnen das Buch *Java für Dummies* (Wiley-VCH). Wenn Sie erst einmal die Java-Grundlagen beherrschen, sollten Sie auf das vorliegende Buch besser vorbereitet sein.

Auch wenn das Betriebssystem Android hauptsächlich aus Java-Code besteht, gibt es einige kleinere Teile des Frameworks, die nicht in Java geschrieben sind. Android-Apps verwenden neben Java auch kleine XML-Abschnitte. Daher sollten Ihnen für das vorliegende Buch auch die XML-Grundlagen halbwegs vertraut sein.

Java: Ihre Android-Programmiersprache

Android-Apps werden in Java geschrieben. Dabei handelt es sich zwar nicht um das ausgewachsene Java, an das die Entwickler für die Java-Unternehmensplattform (JEE – Java Platform, Enterprise Edition) gewöhnt sind, aber doch um eine für Android sehr praktische Java-Untermenge, die einige Klassen ausspart, die für Mobilgeräte sinnlos sind. Wenn Sie bereits Erfahrungen in Java gesammelt haben, sollten Sie sich im Bereich der App-Entwicklung für Android direkt heimisch fühlen.

Selbst wenn Sie kein Java-Referenzhandbuch griffbereit halten, können Sie, wenn Sie bestimmte Befehle nicht ganz verstehen, immer noch danach googeln, beispielsweise unter `www.google.com` oder `www.stackoverflow.com`. Da Java keineswegs neu ist, finden Sie für nahezu alle Einsatzzwecke eine Menge Beispiele im Web.

Nicht alle Klassen, die Java-Programmierer nutzen können, sind auch unter Android verfügbar. Prüfen Sie vor deren Nutzung, ob sie bereitstehen. Falls nicht, gibt es aber wahrscheinlich für Ihre Zwecke passende Alternativen in den Android-APIs.

Activities

Android-Apps bestehen aus einer oder mehreren *Activities*. Ihre Android-App muss mindestens eine Activity enthalten, kann aber auch mehrere umfassen. Eine Activity ist eine Art Container für Teile der Benutzeroberfläche und den für deren Ausführung benötigten Code. Sie können sich Activities als einzelne Seiten Ihrer App vorstellen – eine Seite entspricht einer Activity. Activities werden in den Kapiteln 3 und 5 ausführlicher behandelt.

Fragmente

Bei den einzelnen »Seiten« einer Android-App handelt es sich um jeweils eigenständige Activities. In älteren Android-Versionen haben Sie ein Element, das Sie auf dem Bildschirm anzeigen wollten, direkt in der `Activity`-Klasse abgelegt. Diese Vorgehensweise funktionierte zufriedenstellend bei kleinen Smartphonebildschirmen, auf denen typischerweise nicht sonderlich viele Informationen gleichzeitig dargestellt werden können. Darauf lässt sich vielleicht eine Aufgabenliste oder ein zu bearbeitender Text anzeigen, beide Elemente gleichzeitig auf diesen kleinen Bildschirmen darzustellen, ist aber unmöglich.

Auf den Bildschirmen von Tablets haben Sie hingegen massig Platz. Hier ist es nicht nur sinnvoll, Benutzern eine Aufgabenliste auf dem Bildschirm anzuzeigen und sie diese auf derselben Seite bearbeiten zu lassen, es würde vielmehr sogar recht umständlich wirken, wenn man anders vorgehen würde. Die Bildschirme der meisten Tablets bieten einfach zu viel Platz, um sie mit einzelnen langen Listen oder einer Menge Leerraum zu füllen.

Unter Android lassen sich zwei Activities nicht so einfach gleichzeitig auf den Bildschirm bringen. Was also ist zu tun? *Fragmente* sind die Lösung.

Wenn Sie Fragmente benutzen, kann eine Liste die eine Hälfte des Bildschirms belegen, während der Bearbeitungsbereich die andere Hälfte benutzt. Wie Sie Fragmente in Ihrer Smartphoneanwendung nutzen können, werden Sie in Kapitel 9 erfahren. Wie Sie bei der Skalierung Ihrer App für Tablets vorgehen können, erfahren Sie dann in Kapitel 17.

Fragmente können Sie sich als Mini-Activities vorstellen: Da alle Fragmente einen eigenen Lebenszyklus besitzen, wissen Sie unter anderem, wann sie erzeugt und zerstört werden. Fragmente werden in Aktivitäten eingebettet.

Intents

Intents bilden den Kern des Nachrichtensystems von Android. Intents bestehen aus zwei Elementen:

- ✔ **Einer Aktion.** Die allgemeine Aufgabe (Anzeigen, Bearbeiten, Wählen und so weiter), die nach dem Empfang eines Intents ausgeführt werden soll.
- ✔ **Daten.** Die Daten, die von einer Aktion verarbeitet werden sollen, wie beispielsweise der Name eines Kontakts.

Intents werden zum Starten von Aktivitäten und für die Kommunikation zwischen den verschiedenen Teilen des Android-Systems genutzt. Applikationen können Intents senden und empfangen.

Messages mit Intents versenden

Wenn Sie einen Intent senden, handelt es sich dabei um eine *Message* (Systemnachricht), die Android mitteilt, dass es etwas geschehen lassen soll. Ein Intent könnte Android aus Ihrer App heraus zum Starten einer neuen Activity oder einer anderen App bewegen.

Intent-Filter registrieren

Nur weil Sie einen Intent versenden, muss noch lange nicht automatisch etwas passieren. Sie müssen einen *Intent-Filter* registrieren, der auf die Intents wartet und Android dann mitteilt, was es tun soll und ob die Task eine neue Activity oder eine andere App starten will. Wenn mehrere Empfänger (Receiver) mit einem bestimmten Intent etwas anzufangen wissen, kann ein *Chooser* (Auswahlmenü) erstellt werden, über den der Anwender entscheiden kann, mit welcher App eine Activity weiterverarbeitet werden soll. Ein Beispiel dafür ist die YouTube-App, die den Anwender entscheiden lässt, ob er Videos in der YouTube-App oder in einem Browser anzeigen will.

Verschiedene registrierte Receiver wie beispielsweise die Gmail- oder Hangouts-Apps wissen standardmäßig, was mit Intents zum Teilen von Bildern zu tun ist. Wenn es mehrere mögliche Intent-Filter gibt, wird dem Anwender ein Auswahlmenü angezeigt, in dem er gefragt wird, wie weiter vorgegangen werden soll: das E-Mail-Messaging verwenden oder eine andere Applikation, wie in Abbildung 1.1 gezeigt.

Abbildung 1.1: Ein Auswahlmenü (Chooser)

Wenn das Android-System keinen passenden Empfänger für einen gesendeten Intent findet und manuell kein Chooser erzeugt wurde, stürzt die App zur Laufzeit mit einem *Ausnahmefehler* (*run-time exception*) ab und es kommt zu einem nicht behebbaren Fehler in der App. Android erwartet von Entwicklern, dass sie wissen, was sie tun. Weitere Informationen über die Verwendung von Intent-Choosern finden Sie unter `http://d.android.com/training/basics/intents/sending.html`.

Cursorlose Steuerelemente

Anders als beim PC, bei dem Sie einen Zeiger mit der Maus über den Bildschirm schubsen, ersetzen Ihre Finger bei Android-Geräten nahezu alle Funktionen einer Maus. Und was ist mit dem Rechtsklick? Bei Android wird der Rechtsklick durch den Langdruck ersetzt. Wenn Sie Ihren Finger über längere Zeit hinweg auf eine Schaltfläche (auch *Knopf* für Linux-Anwender), ein Symbol oder den Bildschirm legen, wird ein Kontextmenü angezeigt.

Als Entwickler können Sie Kontextmenüs erstellen und ändern. Zudem können Sie bei Android-Geräten beispielsweise zwei Finger anstelle nur eines Mauszeigers verwenden. Denken Sie aber daran, dass Finger unterschiedlich groß sind, und entwerfen Sie Ihre Benutzeroberflächen entsprechend. Die Schaltflächen sollten groß genug sein und ausreichend Abstand untereinander haben, damit selbst Anwender mit riesigen Pranken Ihre Apps auf kleinen Smartphones und Tablets leicht bedienen können.

Views

Was ist denn das nun schon wieder? Bei einer *View* handelt es sich um ein Grundelement der Android-Benutzeroberfläche, einen rechteckigen Bereich auf dem Bildschirm, der Objekte anzeigt und Ereignisse verarbeitet. Views sind also grundlegende Bausteine von Android-Benutzeroberflächen, ganz ähnlich wie Absatz- oder Anker-Tags (<p> und <a>) Bausteine einer HTML-Seite sind. Einige Beispiele für Views, die in Android-Apps verwendet werden können, sind `TextView`, `ImageView`, `Layout` und `Button`, es gibt jedoch noch unzählige andere, die Sie auch unbedingt kennenlernen sollten. Und Sie können auch Ihre eigenen, benutzerdefinierten Views implementieren.

Es gibt also noch viel mehr Views, die auf Sie warten. Einzelheiten dazu finden Sie bei den Paketen `android.widget` und `android.view` in der Android-Dokumentation unter `http://developer.android.com/reference/packages.html`.

Hintergrundoperationen

In Android gibt es verschiedene Möglichkeiten, mehrere Operationen gleichzeitig ausführen zu lassen, ohne selbst Threads verwalten zu müssen (was jedoch generell nicht zu empfehlen ist). Wenn Sie Daten aus einer Datenbank laden, um diese auf dem Bildschirm anzuzeigen, verwenden Sie im Allgemeinen *Loader*. Loader übernehmen die Verwaltung der Hintergrund-Threads für Sie und überwachen außerdem Ihre Datenbank auf Änderungen, sodass Ihre Benutzeroberfläche aktualisiert wird, sobald sich Daten ändern. Weitere Informationen über Loader finden Sie in Kapitel 13.

Für andere Hintergrundoperationen könnten Sie beispielsweise die Klasse `AsyncTask` verwenden, um eine Operation als Hintergrund-Thread auszuführen. Mit `AsyncTask` können Sie einen Task im Hintergrund starten und sein Ergebnis an Ihren Vordergrund-Thread weitergeben, sodass Sie Ihre Benutzeroberfläche aktualisieren können. Damit erhalten Sie ein sauberes Programmiermodell für die asynchrone Verarbeitung.

Mithilfe von *Threads* können Sie auf einem Gerät mehrere Anweisungsmengen gleichzeitig ausführen. Sie verwenden alle denselben Speicher und dieselbe CPU, aber wenn ein Thread unterbrochen wird, weil er beispielsweise auf etwas anderes wartet, können andere Threads fortgesetzt werden, damit die CPU immer etwas zu tun hat.

Die asynchrone Verarbeitung sollten Sie für Tasks nutzen, deren Ausführung länger als nur ein paar Sekundenbruchteile dauert, also beispielsweise für die Netzwerkkommunikation (Internet) oder das Lesen/Schreiben von Daten auf Speichermedien. Wenn die Anwender auf die Ergebnisse Ihres Tasks warten müssen, sollten Sie einen asynchronen Aufruf und ein Element auf der Benutzeroberfläche verwenden, um ihnen mitzuteilen, dass gerade irgendetwas passiert.

Wenn Sie kein asynchrones Programmiermodell nutzen, könnten Anwender Ihre App (berechtigterweise) für fehlerhaft halten. Es dauert beispielsweise ein wenig, die neuesten Twitter-Mitteilungen aus dem Internet herunterzuladen. Wenn das Netzwerk stark ausgelastet ist und Ihre App nicht asynchron arbeitet, hängt sie scheinbar und reagiert nicht mehr auf Eingaben, woraufhin Anwender meinen könnten, dass ein Fehler aufgetreten ist. Wenn eine App nicht innerhalb eines an-

gemessenen Zeitrahmens reagiert, wird ein Dialogfeld angezeigt, das den Anwender darüber informiert, dass die App nicht mehr reagiert (siehe Abbildung 1.2). Dann kann der Anwender entscheiden, ob er die App schließen oder auf ihre Wiederherstellung warten will. Größtenteils klicken die Benutzer auf OK und schließen Ihre App.

Abbildung 1.2: Hier wird gemeldet, dass eine App nicht mehr reagiert.

Am besten betten Sie CPU-lastigen oder länger laufenden Code in einen anderen Thread ein. Die Vorgehensweise wird auf der Seite *Keeping Your App Responsive* auf der Website für Android-Entwickler unter `http://d.android.com/guide/practices/design/responsiveness.html` beschrieben.

Hintergrunddienste

Sie wissen möglicherweise, was ein *Dienst* (*Service*) ist, nämlich eine im Hintergrund laufende Anwendung, die nicht notwendigerweise eine Benutzeroberfläche besitzen muss. Klassische Beispiele sind die üblicherweise als Hintergrunddienst laufenden Antivirenprogramme. Auch wenn sie auf dem Bildschirm unsichtbar bleiben, wissen Sie doch, dass sie laufen.

Auch Android-Apps können Hintergrunddienste verwenden. Die meisten als Download im Google Play Store erhältlichen Audioplayer können Sie als Hintergrunddienst laufen lassen. Auf diese Weise können Sie beim Abruf Ihrer E-Mails oder der Erledigung anderer Aufgaben, für die der Bildschirm benötigt wird, weiter Musik hören.

Android Support Library

Natürlich macht es am meisten Spaß, Apps für die neuesten und leistungsstärksten Geräte zu entwickeln! Es kommt jedoch auch immer wieder vor, dass Sie ältere Geräte unterstützen müssen. Schließlich verwenden nicht alle Ihre Anwender garantiert immer die allerneueste Version von Android.

Glücklicherweise bietet Android aber auch hier eine Lösung. Sie können nämlich die *Android Support Library* benutzen, um Ihre Apps mit allen Geräten bis zurück in die Android-Steinzeit (etwa 2010 nach Christus) kompatibel zu machen.

Diese Hilfsbibliotheken unterstützen nicht nur Fragmente und Loader, sondern erweitern die alten Geräte auch um einige neuere APIs wie:

- ✔ **RecyclerView**: Erzeugt eine endlos blätterbare View-Liste.
- ✔ **CardView**: Eine »Karte«, die Sie mit einer `RecyclerView` nutzen können, um eine blätterbare Liste mit Karten in Ihren Apps zu erstellen.

- ✔ **ViewPager:** Seiten nach links und rechts wischen.
- ✔ **ShareCompat:** Zum Teilen aller möglichen Dinge mit Freunden.

Eine vollständige Liste der Funktionen der Android Support Library finden Sie unter `https://developer.android.com/tools/support-library/features.html`. Unter `https://developer.android.com/about/dashboards` erfahren Sie, wie viele Android-Benutzer welche Android-Versionen verwenden.

Aktionsleiste

In der *Aktionsleiste* bringen Sie zahlreiche Schaltflächen und Menüs unter, die den Anwendern gestatten, mit Ihrer App zu interagieren. Die Aktionsleiste wird fast immer oben quer über den Bildschirm angezeigt – man kann sie also *kaum* übersehen. Abbildung 1.3 zeigt ein Beispiel für die Aktionsleiste der YouTube-App.

Abbildung 1.3: Die YouTube-Aktionsleiste für ein Android-Demovideo

Beachten Sie diese Elemente in der Aktionsleiste:

- ✔ **Aufwärts-Schaltfläche, App-Logo:** Tippen Sie das App-Logo in der Aktionsleiste an, um eine Ebene aufwärtszugehen.

Beachten Sie den feinen Unterschied zwischen der Aufwärts- und der Zurück-Schaltfläche. Über die Zurück-Schaltfläche kehrt der Benutzer zur vorherigen Aktivität zurück, wobei die benutzte App keine Rolle spielt. Die Aufwärts-Schaltfläche bringt den Benutzer hingegen zurück zur letzten Aktivität *in der aktuellen Anwendung*, selbst wenn es sich bei dieser nicht um die Aktivität handelt, die er zuletzt genutzt hat.
Angenommen, Sie betrachten eine Webseite im Chrome-Browser und tippen einen Link an, um die YouTube-App zu starten. Wenn Sie dann die Zurück-Schaltfläche antippen, kehren Sie damit zu Chrome zurück. Über die Aufwärts-Schaltfläche gelangen Sie hingegen zur Startseite der YouTube-App.

- ✔ **Seitentitel:** Neben dem Symbol der Anwendung wird der Titel der aktuellen Seite in der Aktionsleiste angezeigt. Wenn sich auf der aktuellen Seite Ihrer App die Daten irgendwie filtern lassen, können Sie dort ein Dropdown-Menü einfügen, über das sich der Filter wechseln lässt.
- ✔ **Aktion:** Am rechten Ende der Aktionsleiste können Sie verschiedene Aktionen sehen, die Benutzer ausführen können. In der YouTube-App in Abbildung 1.4 können Benutzer Videos zu einer Liste hinzufügen, sie weiterempfehlen oder nach weiteren Videos suchen. Für die Aktionsschaltflächen können Texte und/oder Symbole angezeigt werden. Sie können beliebig viele Aktionen hinzufügen. Aktionen, für die der Platz auf dem Bildschirm nicht mehr ausreicht, werden ganz rechts in ein Untermenü ausgelagert.

- ✔ **Kontext-Aktionsleiste (nicht gezeigt):** Das Aussehen der Aktionsleiste kann geändert und an die jeweiligen Aktivitäten der Benutzer angepasst werden. Werden beispielsweise mehrere Elemente in einer Liste ausgewählt, können Sie die Standard-Aktionsleiste durch eine *kontextbezogene* Aktionsleiste ersetzen, um Benutzern die Auswahl auf der Grundlage dieser Elemente zu ermöglichen. Wenn beispielsweise mehrere Dateien gleichzeitig gelöscht werden können sollen, können Sie eine Löschen-Schaltfläche in die Kontext-Aktionsleiste aufnehmen.

Unter `http://d.android.com/guide/topics/ui/actionbar.html` finden Sie weitere Informationen über die vielseitigen Möglichkeiten, die dieses Element der Benutzeroberfläche in Ihren Apps bieten kann.

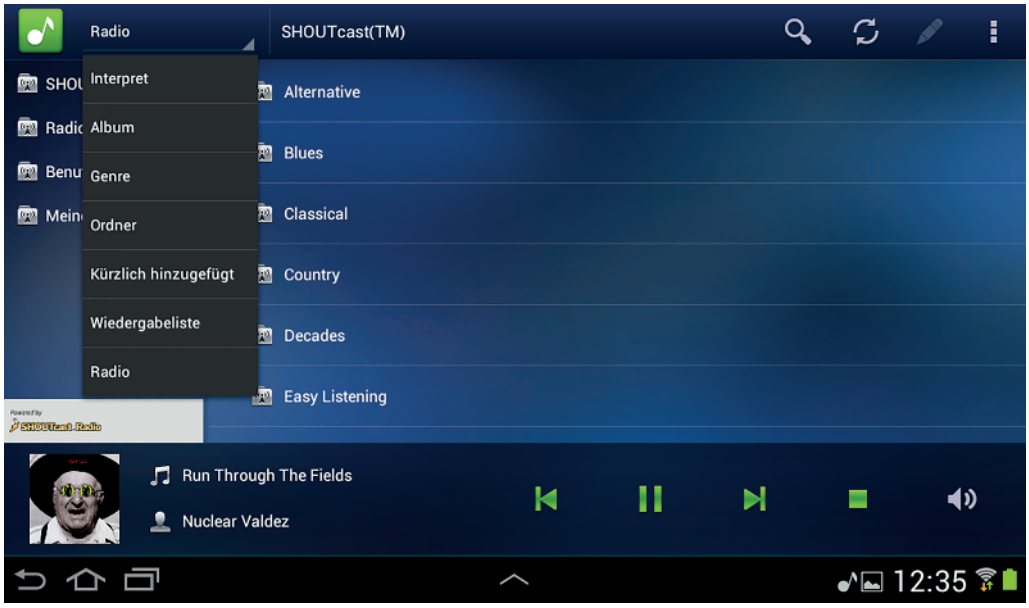

Abbildung 1.4: Audio-App mit Menü in der Aktionsleiste

Unter Android 2.x und in noch älteren Versionen existiert die Aktionsleiste schlichtweg nicht! In diesen Android-Versionen werden in Ihren Apps enthaltene Aktionsleisten nicht angezeigt. Sie können jedoch die Aktionsleiste auch in 2.1 oder früher mithilfe der Support Library nutzen.

Widgets und Benachrichtigungen

Möglicherweise wollen die Anwender auf Informationen Ihrer App zugreifen, ohne diese explizit zu starten. Denken Sie beispielsweise an die Gmail-App, die es den Anwendern ermöglicht, eine Vorschau auf ihre E-Mails in einer Benachrichtigung anzuzeigen, bevor sie die Gmail-App öffnen, oder wie Sie die aktuelle Zeit auf Ihrem Launcher anzeigen, ohne dass Sie die Uhr-

App öffnen müssen. Dies sind Beispiele für die Verwendung von Benachrichtigungen und Launcher-Widgets.

- ✔ **Launcher-Widgets:** Widgets sind eine Art »Mini-Apps«, die Zugriff auf die Funktionalität Ihrer App direkt vom Launcher des Smartphones (auch als Startbildschirm bezeichnet) aus ermöglichen. Widgets sind in der Applikationsliste leicht zu finden. Sie können Informationen anzeigen, Schaltflächen und sogar *Listenansichten* enthalten, innerhalb derer ein begrenztes Wischen und Blättern möglich ist.
- ✔ **Benachrichtigungen:** Benachrichtigungen in Android können erweitert und verkleinert werden, sodass mehr Informationen angezeigt werden können. Wenn Ihnen beispielsweise Ihre Mutter ein Foto ihres neuen Hündchens in einer Textnachricht zuschickt, lässt es sich direkt in der Benachrichtigung anzeigen, ohne dass dazu erst die App geöffnet werden müsste. In einer Benachrichtigung über eine neue E-Mail-Nachricht kann eine Vorschau des Nachrichtentextes angezeigt werden, sodass er direkt gelesen werden kann.

  Zudem können Benutzer unabhängig von der benutzten App direkt auf Benachrichtigungen reagieren. Um beispielsweise auf eine Geburtstags-E-Mail Ihrer Oma zu antworten, müssen Sie nur noch »Antworten« in der Benachrichtigung antippen, um Gmail samt Editor zu starten und sich bei ihr bedanken zu können.

Hardwarefunktionen

Google hat eine Unmenge an Funktionen in Android integriert und stellt Entwicklern alle benötigten Werkzeuge zur Verfügung, um hervorragende und umfassende Apps für Mobilgeräte entwickeln zu können. Google macht es Ihnen leicht, die Funktionen der verfügbaren Hardwarekomponenten zu nutzen.

Um eine spektakuläre Android-App zu entwickeln, sollten Sie die Vorteile der verfügbaren Hardwarekomponenten nutzen. Aber verstehen Sie mich hier nicht falsch: Wenn Sie eine Idee für eine App haben, die ohne Hardwareunterstützung auskommt, dann ist auch das völlig in Ordnung.

Android-Geräte sind mit verschiedenen Hardwarekomponenten ausgestattet, die Sie im Rahmen Ihrer Apps nutzen können und die in Tabelle 1.1 im Überblick dargestellt werden.

Hardwarekomponente	Funktion
Beschleunigungssensor	Registriert Bewegungen des Smartphones
Bluetooth-Funk	Ermöglicht beispielsweise den Anschluss eines Bluetooth-Kopfhörers oder einer externen Tastatur
Kompass	Stellt fest, in welche Richtung das Gerät zeigt
Kamera	Nimmt Fotos oder Videoclips auf
GPS-Empfänger	Ermittelt, wo sich das Gerät befindet

Tabelle 1.1: Android-Gerätehardware

Die meisten Android-Geräte sind zwar mit den in den folgenden Abschnitten beschriebenen Komponenten ausgestattet, deren Leistungsdaten unterscheiden sich aber zum Teil erheblich. Da Android von Hardwareherstellern kostenlos vertrieben werden darf, wird es für eine breite Gerätepalette genutzt. Bei kleineren Herstellern kann daher bei den Geräten schon einmal die eine oder andere Komponente fehlen oder auch zusätzlich hinzukommen.

Android-Geräte sind in den verschiedensten Formen und Abmessungen erhältlich, wie zum Beispiel als Mobiltelefon, Tablet-PC, E-Book-Reader, Uhren, Fernsehgeräte und Autos. Die Android-Entwickler stellen Werkzeuge bereit, mit denen Sie leicht Apps für unterschiedlich dimensionierte und auflösende Bildschirme erstellen können. Und da Ihnen die Android-Entwickler bereits die wirklich schweren Aufgaben abnehmen, brauchen Sie sich auch in dieser Hinsicht keine Sorgen zu machen. Die Grundlagen der Unterstützung verschiedener Bildschirmabmessungen und Auflösungen finden Sie in Kapitel 4.

Touchscreen

Android-Mobiltelefone werden über Touchscreens bedient, die weitreichende Möglichkeiten zur Verbesserung der Bedienung der Apps bieten. Anwender können beispielsweise durch Antippen, Drehen, Ziehen, Wischen und Drücken mit einem oder mehreren Fingern ein Bild auswählen und vergrößern. Entwickler können in Apps sogar angepasste Gesten definieren, um sich noch mehr Möglichkeiten zu erschließen.

Android unterstützt auch *Multitouch-Gesten*, bei denen der gesamte Bildschirm gleichzeitig mit mehreren Fingern berührt wird.

Echte Schalter an Geräten sind nichts Neues. An vielen Android-Geräten (insbesondere Tablets) finden Sie aber neben einem Standby-Schalter nur noch einen Lautstärkeregler. Wenn Sie eine geeignete Benutzeroberfläche für Ihre App erstellen, können Sie beliebig aussehende Schaltflächen an einer beliebigen Stelle auf dem Bildschirm positionieren.

GPS

Wenn das Android-Betriebssystem mit dem GPS-Empfänger eines Geräts zusammenarbeitet, können entsprechend entwickelte Apps jederzeit den aktuellen Standort eines Anwenders feststellen. Sie können auch dessen Bewegungen und Ortswechsel aufzeichnen.

Ein weiteres nützliches Beispiel ist die Maps-App, mit der die Position des Anwenders auf einer Karte bestimmt und Wegbeschreibungen zu einem angegebenen Ziel angezeigt werden können. Mit Android und der GPS-Hardware können Sie auf einige Meter genau die jeweilige Geräteposition feststellen. Viele Apps nutzen diese Funktion, um angeben zu können, wo sich Tankstellen, Restaurants, Parkhäuser, öffentliche Toiletten oder andere interessante Einrichtungen befinden.

Beschleunigungssensor

Android unterstützt einen *Beschleunigungssensor* (*Accelerometer*), mit dem ermittelt werden kann, ob das Gerät bewegt oder geschüttelt wird und in welche Richtung es gedreht wird. Diese Informationen können Sie zur Steuerung Ihrer Apps nutzen.

Damit lässt sich beispielsweise feststellen, ob der Bildschirm gerade auf dem Kopf steht und Ihre App darauf reagieren sollte. Oder vielleicht wollen Sie ein Würfelspiel entwickeln, bei dem Ihre Anwender ihr Smartphone zum Wurf schütteln müssen. Durch derartige Funktionen heben sich Mobilgeräte von typischen Desktop-PCs ab.

Android verfügt über eine eingebaute Aktivitätserkennung, wofür verschiedene Sensoren verwendet werden, wie beispielsweise der Beschleunigungssensor und das GPS, um festzustellen, ob Ihr Anwender gerade geht, läuft, fährt oder mit dem Rad fährt. Weitere Informationen über die Verwendung der Aktivitätserkennung finden Sie unter `http://d.android.com/training/location/activity-recognition.html`.

SD-Speicherkarte

Android sorgt für alle für den Zugriff (Speichern und Laden von Dateien) auf die *SD-Speicherkarte* erforderlichen Funktionen. Bei vielen Mobiltelefonen und Computern lassen sich SD-Speicherkarten einsetzen und zur Speicherung von Daten nutzen. Wenn ein Gerät mit einer SD-Speicherkarte ausgerüstet ist, können Sie darauf alle von Ihrer App benötigten Dateien speichern und darauf zugreifen. Um den internen Gerätespeicher nicht unnötig für benötigte Zusatzressourcen zu verschwenden, können Sie diese oft auch ganz oder teilweise von einem Webhost herunterladen und auf SD-Karte speichern. Benutzer finden dies vielleicht besser und werden Ihre App dann nicht so schnell deinstallieren, um Speicherplatz für andere Apps freizumachen.

SD-Speicherkarten sind zwar bei vielen, aber längst nicht allen Geräten auch wirklich vorinstalliert. Bevor Sie versuchen, in Ihren Apps Daten auf SD-Karte zu speichern, sollten Sie daher zunächst immer erst einmal prüfen, ob eine SD-Karte installiert ist und ob darauf noch genug Platz für die zu speichernden Dateien vorhanden ist. Denken Sie außerdem daran, dass Dateien auf einer SD-Speicherkarte nicht sicher sind und von allen anderen Apps auf dem Smartphone des Anwenders gelesen werden können.

Softwarewerkzeuge

Beim Schreiben von Android-Apps stehen Ihnen verschiedene Werkzeuge zur Verfügung. In den folgenden Abschnitten werde ich Ihnen einige der beliebtesten Tools kurz vorstellen, die Sie im Rahmen Ihrer alltäglichen Android-Entwicklungsarbeit nutzen werden.

Internet

Dank der Internetfähigkeit von Android-Geräten lassen sich aktuelle Daten leicht abrufen. Als Anwender können Sie das Internet nutzen, um sich über die Anfangszeit des nächsten Films oder die Ankunftszeit der nächsten Straßenbahn zu informieren. Als Entwickler können Sie das Internet in Ihren Apps für den Zugriff auf aktuelle Wetterdaten, Nachrichten und Sportergebnisse nutzen. Sie können das Web aber (wie Pandora und YouTube) auch dazu benutzen, um dort die Symbole und Abbildungen Ihrer Apps zu speichern.

Lassen Sie es nicht damit bewenden. Falls sinnvoll, können Sie auch Teile Ihrer App auf einen Webserver auslagern! Dadurch lässt sich manchmal sogar viel Verarbeitungszeit einsparen und dafür sorgen, dass Ihre Android-App schlank bleibt. Diese Vorgehensweise wird *Client-Server-Computing* genannt. Bei dieser verbreiteten Softwarearchitektur richtet der Client Anforderungen an einen Server, der nur darauf wartet, etwas zu tun zu bekommen. Die integrierte Maps-App ist ein gutes Beispiel für eine Anwendung, bei der ein Client auf die auf einem Webserver abgelegten Karten und Standortdaten zurückgreift.

Audio- und Videounterstützung

Mit dem Android-Betriebssystem lässt sich in Ihren Apps auch Audio und Video spielend leicht nutzen. Dabei werden viele Standardformate unterstützt. Das Einbinden multimedialer Inhalte in Ihre Apps könnte kaum einfacher sein. Toneffekte, Anleitungsvideos, Hintergrundmusik, Videostreams und Audio aus dem Internet lassen sich problemlos zu Ihren Apps hinzufügen. Lassen Sie Ihrer Kreativität freien Lauf. Ihren Möglichkeiten sind kaum Grenzen gesetzt.

Kontakte

Ihre App kann auf die auf dem Gerät gespeicherten Kontaktdaten zugreifen. Diese Funktion können Sie nutzen, um Kontakte auf neue Weise oder einfach anders anzuzeigen. Vielleicht gefällt Ihnen die bereits vorhandene App zur Verwaltung der Kontakte nicht. Da Sie auf die auf dem Gerät gespeicherten Kontakte zurückgreifen können, hält Sie nichts davon ab, sich Ihre eigene App zu schreiben. Vielleicht kombiniert diese ja auch Ihre Kontakte mit dem GPS-System und benachrichtigt den Anwender, wenn er sich in der Nähe einer der Kontaktadressen befindet.

Nutzen Sie Ihre Fantasie, gehen Sie dabei aber verantwortungsvoll vor. Keinesfalls sollten Sie Kontaktdaten missbrauchen (siehe nächster Abschnitt).

Sicherheit

Stellen Sie sich nur einmal vor, dass jemand eine App veröffentlicht, die Kontaktlisten durchsucht und sie komplett irgendwo auf einen Server überträgt, um die Daten dann für eigene Zwecke zu missbrauchen. Anhand dieses Gedankens sollte Ihnen klar geworden sein, warum den meisten Funktionen, die die Gerätekonfiguration ändern oder auf geschützte Inhalte zugreifen, speziell entsprechende *Berechtigungen* eingeräumt werden müssen. Angenommen, Sie wollen ein Bild aus dem Web herunterladen und auf der SD-Karte speichern. Dazu benötigen Sie eine Berechtigung für den Zugriff auf das Internet. Sie benötigen auch entsprechende Berechtigungen, um Dateien auf der SD-Karte speichern zu können. Zu Beginn der Installation einer App wird deren Benutzer darüber informiert, welche Berechtigungen sie erfordert. Dann kann er entscheiden, ob die Installation fortgesetzt werden soll. Um eine Berechtigung anzufordern, müssen Sie nur eine Zeile Code in die Manifest-Datei Ihrer App einfügen. (Manifest-Dateien werden in Kapitel 3 beschrieben.)

Google-APIs

Das Android-Betriebssystem lässt Sie nicht nur telefonieren, Kontakte verwalten oder Apps installieren. Es bietet weit mehr. Als Entwickler bieten sich Ihnen weitreichende Möglichkeiten. Sie können beispielsweise sogar Landkarten in Ihre Apps integrieren. Dazu brauchen Sie nur die Google-Maps-API.

Positionen in Landkarten kennzeichnen

Vielleicht wollen Sie eine App schreiben, die Ihren Freunden Ihren aktuellen Aufenthaltsort mitteilt. Für die Entwicklung eines Kartensystems könnten Sie entweder Hunderte Stunden Entwicklungszeit aufwenden – oder einfach die Google-Maps-API nutzen. Sie können die API einbinden und jede Menge Entwicklungszeit sparen, ohne dass es Sie auch nur einen einzigen Cent kosten würde. Mit der Maps-API können Sie fast alles nur über eine Adresse finden. Die Möglichkeiten sind schier grenzenlos. Zeigen Sie den Aufenthaltsort Ihrer Freundin, die nächste Bäckerei, die nächste Tankstelle oder andere Dinge über deren Adresse an.

Es ist zwar nett, Ihre Freunde über Ihren aktuellen Aufenthaltsort zu informieren, aber das ist noch längst nicht alles! Die Google-Maps-API kann auch auf Googles Navigation-API zugreifen. Und schon können Sie Ihren Anwendern nicht nur Ihre aktuelle Position, sondern auch den Weg dorthin anzeigen.

Cloud-Messaging

Noch so ein obskurer Begriff! Beim Cloud-Computing befinden sich die Rechner, die irgendwelche Dienste bereitstellen, irgendwo in den »Wolken« (clouds), ohne dass sich ihr Standort (im Internet) genau ausmachen ließe. Sie könnten die Daten Ihrer App beispielsweise irgendwo in der Wolke (dem Internet) speichern und die benötigten Daten beim ersten Starten Ihrer App herunterladen. Was aber, wenn diese sich später ändern? Damit die App aktualisiert werden kann, muss sie darüber benachrichtigt werden. Dazu können Sie Ihrer App beziehungsweise dem Gerät vom Server aus im Web eine Mitteilung zukommen lassen (Cloud-to-Device Message), die dafür sorgt, dass es die aktualisierten Daten herunterlädt. Das funktioniert sogar, wenn Ihre App nicht läuft. Sobald das Gerät die Benachrichtigung erhält, sorgt es dafür, dass Ihre App gestartet wird, und leitet die entsprechenden Maßnahmen ein.

Das KISS-Prinzip

Bei der Entwicklung von Anwendungen kann man es leicht übertreiben und übermäßig komplizierte Lösungen entwickeln. Denken Sie daher an das KISS-Prinzip, das für »Keep It Simple, Stupid« steht und sinngemäß bedeutet, dass man seine Lösungen möglichst einfach halten sollte. Wenn man einfach loslegt, ohne die integrierten APIs wirklich zu verstehen und ohne zu wissen, was sie eigentlich machen, wird der eigene Code leicht übermäßig kompliziert. Sie können zwar so vorgehen, aber das könnte Sie viel mehr Zeit als das Überfliegen der Android-Dokumentation kosten. Sie müssen sich nicht alles merken, sollten sich aber selbst den Gefallen tun und einen Blick in die Dokumentation werfen. Dann wissen Sie, wie leicht Sie die bereits in-

tegrierten Funktionen nutzen und wie viel Entwicklungszeit Sie mit ihnen sparen können. Leicht können Sie viele Zeilen Code schreiben, um etwas zu machen, was sich eigentlich mit einer einzigen Zeile erledigen ließe. Die Lautstärke der Audiowiedergabe zu verändern oder ein Menü zu erstellen, ist zwar eigentlich einfach, aber wenn Sie die APIs nicht kennen, werden Sie vielleicht nur auf weitere Probleme stoßen, wenn Sie versuchen, die entsprechenden Funktionen neu zu schreiben.

Durch Hinzufügen von eigentlich gar nicht benötigten Funktionen kann man eigene Projekte auch wirklich gründlich in den Sand setzen. Sie sollten möglichst die einfachsten Lösungen bevorzugen. Erstellen Sie also keine aufwendigen Dialogfelder mit Registerkarten, wenn es bereits ein paar Menüelemente tun würden. Android verfügt über genug vorgefertigte Steuerelemente (Widgets), mit denen sich nahezu alle Aufgabenstellungen bewältigen lassen. Zudem haben es Benutzer Ihrer App bei den vorgefertigten Widgets mit ihnen bereits vertrauten, geschätzten und daher leicht nutzbaren Bedienelementen zu tun.